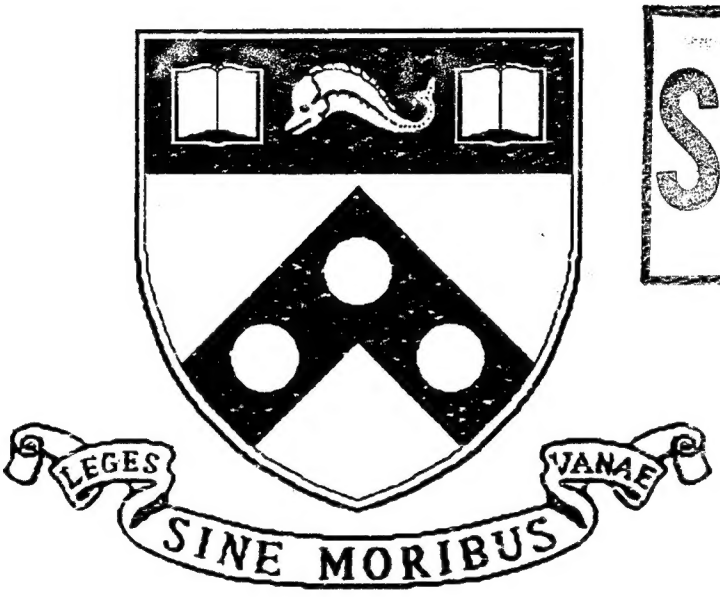
# Simplifying Tool Usage In Teleoperative Tasks

MS-CIS-93-68
GRASP LAB 353

Thomas Lindsay
Richard P. Paul






University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389



July 1993

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | | technical report |

**4. TITLE AND SUBTITLE**
Simplifying Tool Usage In Teleoperative Tasks

**5. FUNDING NUMBERS**
DAAL03-89-C-0031

**6. AUTHOR(S)**
Thomas Lindsay
Richard P. Paul

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
University of Pennsylvania
Department of Computer and Information Sciences
200 S. 33rd Street
Philadelphia, PA 19104-6389

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
ARO 26779.24-MA-AI

**11. SUPPLEMENTARY NOTES**
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

Modern robotic research has presented the opportunity for enhanced teleoperative systems. *Teleprogramming* has been developed for teleoperation in time-delayed environments, but can also lead to increased productivity in non-delayed teleoperation.

Powered tools are used to increase the abilities of the remote manipulator. However, tools add to the complexity of the system, both in terms of control and sensing. Teleprogramming can be used to simplify the operators interaction with the manipulator/tool system. Further, the adaptive sensing algorithm of the remote site system (using an instrumented compliant wrist for feedback) simplifies the sensory requirements of the system. Current remote-site implementation of a teleprogramming tool-usage strategy that simplifies tool use is described in this document.

The use of powered tools in teleoperation tasks is illustrated by two examples, in using an air-powered impact wrench, and the other using an electric winch. Both of these tools are implemented at our remote site workcell, consisting of a Puma 560 robot working on the task of removing the top of a large box.

**14. SUBJECT TERMS**

**15. NUMBER OF PAGES**

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |

# Simplifying tool usage in teleoperative tasks[†]

Thomas Lindsay and Richard P. Paul

GRASP Laboratory, University of Pennsylvania
3401 Walnut Street, Room 301C, Philadelphia, PA 19104-6228

**ABSTRACT**

Modern robotic research has presented the opportunity for enhanced teleoperative systems. *Teleprogramming* has been developed for teleoperation in time-delayed environments, but can also lead to increased productivity in non-delayed teleoperation.

Powered tools are used to increase the abilities of the remote manipulator. However, tools add to the complexity of the system, both in terms of control and sensing. Teleprogramming can be used to simplify the operators interaction with the manipulator/tool system. Further, the adaptive sensing algorithm of the remote site system (using an instrumented compliant wrist for feedback) simplifies the sensory requirements of the system. Current remote-site implementation of a teleprogramming tool-usage strategy that simplifies tool use is described in this document.

The use of powered tools in teleoperation tasks is illustrated by two examples, one using an air-powered impact wrench, and the other using an electric winch. Both of these tools are implemented at our remote site workcell, consisting of a Puma 560 robot working on the task of removing the top of a large box.



[†]Appears in the Proceedings of Telemanipulator Technology, a session of the SPIE/OE Technology '92 Conference, Boston, MA, November 15-20, 1992

# Simplifying tool usage in teleoperative tasks[†]

Thomas Lindsay and Richard P. Paul

GRASP Laboratory, University of Pennsylvania
3401 Walnut Street, Room 301C, Philadelphia, PA 19104-6228

**ABSTRACT**

Modern robotic research has presented the opportunity for enhanced teleoperative systems. *Teleprogramming* has been developed for teleoperation in time-delayed environments, but can also lead to increased productivity in non-delayed teleoperation.

Powered tools are used to increase the abilities of the remote manipulator. However, tools add to the complexity of the system, both in terms of control and sensing. Teleprogramming can be used to simplify the operators interaction with the manipulator/tool system. Further, the adaptive sensing algorithm of the remote site system (using an instrumented compliant wrist for feedback) simplifies the sensory requirements of the system. Current remote-site implementation of a teleprogramming tool-usage strategy that simplifies tool use is described in this document.

The use of powered tools in teleoperation tasks is illustrated by two examples, one using an air-powered impact wrench, and the other using an electric winch. Both of these tools are implemented at our remote site workcell, consisting of a Puma 560 robot working on the task of removing the top of a large box.

## 1 INTRODUCTION

Three major elements of teleprogramming [2, 4] (a form of supervisory control [1]) allow potential improvements over a direct teleoperative system. First, the operator interacts with a virtual remote site, providing opportunities for artificial operator aids, including visual and kinesthetic cues. Second, communication between the master and remote sites is in the form of symbols instead of signals, allowing bandwidth limited and time-delayed teleoperations, as well as increasing the intelligence of communication between master and slave. Finally, the remote site executes successive commands autonomously, allowing the remote site manipulator to interact more intelligently with the environment.

A robot manipulator is greatly limited in its strength to accuracy ratio. For a relatively low-strength robot to execute tasks which require more strength, powered tools may be used in conjunction with the robot. The precision of the low-power robot is coupled with the strength of less-precise heavy tools. For this research, tools include an impact wrench which delivers more torque than the robot, and a winch with more lifting power than the payload capability of the robot.

The tool/manipulator system is complex from the point of control and sensing. Extra degrees of freedom lead to a redundant system, difficult to control and difficult for the operator to operate. Tool usage requires sensing, which may further complicate the system. However, using teleprogramming, a simplified tool control and sensing structure can be implemented.

When a powered tool is selected for use, the teleprogramming command **UseTool(*toolname*)** is sent to the remote site to select parameters and default conditions specific to that tool. A task (control) frame is created wherein the

[†]Appears in the Proceedings of Telemanipulator Technology, a session of the SPIE/OE Technology '92 Conference, Boston, MA, November 15-20, 1992

coordinate axes align with the natural degree(s) of freedom of the tool. The tool is then controlled instead of the corresponding degree of freedom of the robot manipulator, which becomes passive. Control of the tool by the operator is accomplished simply by motion of the master manipulator in the specific degree of freedom which the tool controls.

The addition of sensors for each and every tool would be difficult from an engineering standpoint and would add unnecessary complexity to the system. In our system, the wrist force/torque sensor of the remote robot manipulator is used for tool sensing. General purpose *guarded moves* [5], incorporated in the teleprogramming language, are used to monitor tool actions, and to determine successful tool operation.

The control and sensing structure outlined in this paper is implemented using an impact wrench and a winch. Experimental results are presented.

# 2 CONTROL OF POWERED TOOLS

Powered tools add degrees of freedom to the manipulator/tool system. For example, the impact wrench attached to the end of the manipulator adds a rotational degree of freedom to the system (the natural rotation axis of the wrench). The winch adds a translational degree of freedom in the direction of gravity. These extra degrees of freedom of the system are redundant. Choosing the directions to control and those to become passive is normally an optimization problem. A generalized redundant system would be difficult to control in real time. However, the tool's functionality prescribes that control by the tool of its own natural degree of freedom is necessary for proper usage. This degree of freedom is automatically chosen as a direction to control.

When using a specific tool, a task frame can be assigned that aligns one Cartesian direction of the frame with the natural axis of the tool, and this direction is controlled by the tool. For the robot manipulator, this direction becomes passive, and the system redundancy is removed. Because of the functionality of the powered tool, and the ability to use arbitrary task frames, control of the redundant manipulator/tool system is simplified.

The functionality of the tool is also important in the way the passive degree of freedom is treated. For example, the impact wrench is a force controlled device. The corresponding passive degree of freedom in the manipulator must be position controlled, preventing motion. The winch, on the other hand, is a position controlled device. The corresponding degree of freedom in the manipulator is force controlled with a force preload, so that the manipulator will comply with the motion of the winch and keep the cable taut.

From the human operator's station, the control of the manipulator/tool system requires that a specific tool is chosen for use[1]. After this is accomplished, the operator can move the master arm around and see/feel the tool working in the virtual remote site environment. Motions by the operator in the natural axis of the tool are used to control the tool; no additional actions are needed. Some subtle changes are imposed on the feedback to the master arm in order to aid the operator. or example, control of the impact wrench is based on rotations about the z-axis of the master manipulator's tool frame. In order to prevent unwanted, accidental use of the wrench, a small amount of resistance to motion in this direction is programmed into the master arm controller. Therefore, only deliberate motions to control the tool are accepted as input. The winch control is designed similarly, with a further constraint that the velocity in the z-direction of the base frame is forced to be constant when deliberate motions in this direction are sensed, simulating the constant velocity of the winch. The human operator does not have to specifically control the tools; inputs of motion in the tool direction is all that is needed.

After the tool has been specified for use, commands are automatically generated in the same form as for non-tool actions. Motions, changes in contact states, guards, etc. are in the same form. The remote site, notified that a specific tool is in use, interprets these commands properly for the task.

[1] Tool usage at the master site has not yet been implemented in the experimental testbed.

# 3 SENSING FOR POWERED TOOLS

The addition of a new sensor brings added complexity to the system. It must be calibrated and properly implemented, and the system will depend on the new sensor to work properly. Too many sensors become redundant, and sensor fusion techniques become necessary to correlate sensor inputs. This is expensive in terms of time and computational power, as well as expensive in terms of the physical sensor devices. In order to simplify the problem, and to reduce cost, all tool sensing feedback comes from the existing force/torque sensor of the remote manipulator, the instrumented compliant wrist [3].

The use of sensory inputs for tool control is similar to the implementation in the basic teleprogramming system. Guarded moves terminate properly when expected sensory events are encountered within the tolerance limits. Otherwise, motion is terminated in an error state. Guarded moves are generated automatically by the master system as required, in the same manner as a motion without tool usage. An adaptive sensing algorithm is used to determine sensor events from normal operating noise, which changes with different tools and operating conditions. Therefore, the same algorithm can be used with or without tools in the system.

# 4 EXPERIMENTAL RESULTS

## 4.1 Impact wrench

An impact wrench has advantages over a conventional wrench used by the robot. It can deliver much greater torque than the robot can by itself. The impact wrench is easier to use than the conventional wrench, because no complex movements are needed (however, most robot systems that use wrenches use electric or pneumatic wrenches which also do not require motion of the robot). Drawbacks to using the impact wrench include vibrations caused by the tool, and the need for a power source. The use of the impact wrench is characteristic of other tools the robot may use, such as drills and screwdrivers, which have the same natural axis of motion.

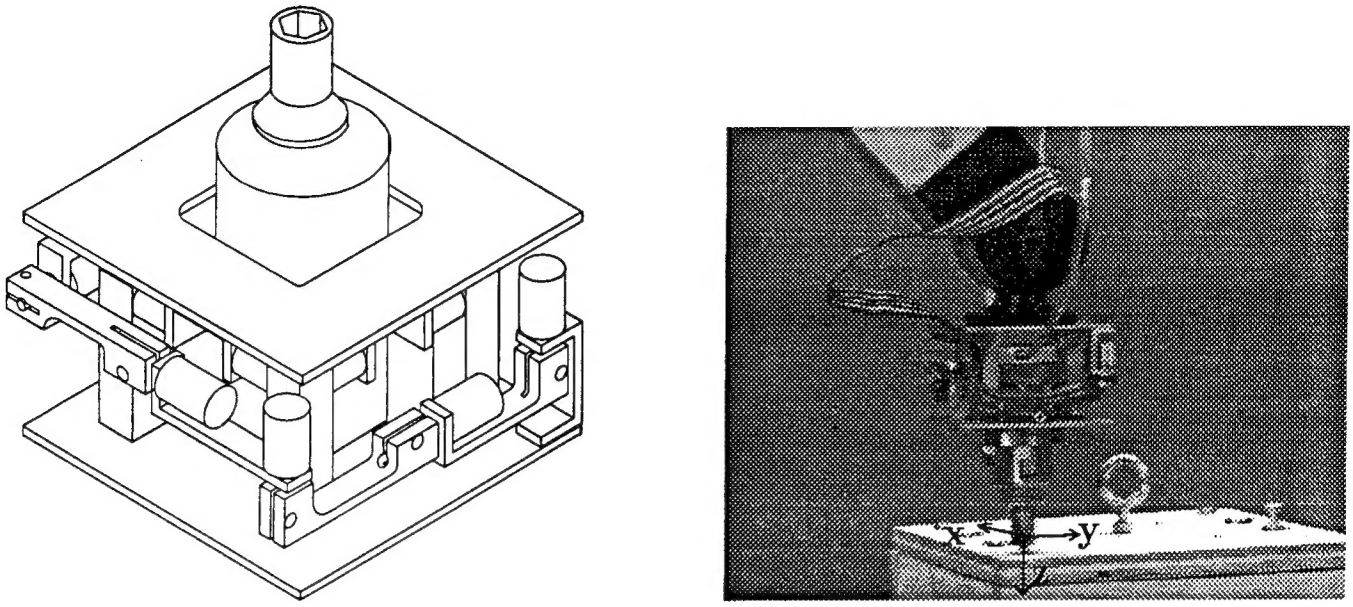
The impact wrench used for this research is a 3/8" drive pneumatic wrench. Mounted internally on the instrumented compliant wrist (see figure 1(a)), it creates an extra degree of freedom with the same axis of rotation as the z-axis of the end effector frame (T6). By specifying the wrench in the UseTool() command, rotation about the z-axis of T6 implies control of the wrench. The corresponding z-axis in the manipulator is position controlled with no specified motion, and thus acts as a locked joint (no motion allowed).

Inserting a bolt involves either sensing torques about the axis of rotation, or sensing velocity in the bolt feed direction. A high torque, or zero feed velocity, may indicate a jammed bolt or a properly seated bolt, so tolerances must be checked. Removing a bolt requires that the outward velocity be monitored. When the outward velocity goes to zero, the bolt is assumed to be fully extracted.

When the impact wrench is specified for use, the following parameters are automatically assumed:

- The z-rotation mode for the robot manipulator is set for position control. This assumes that the current task frame is aligned with the tool frame of the manipulator ($T6$).
- Gains and maximum allowable forces/velocities are changed to reflect the increase in noise when the tool is in use.
- Motions in the positive z-rotation direction are transformed to turn the impact wrench clockwise. Similarly, motions in the negative z-rotation direction turn the impact wrench on counterclockwise.

The following is a sample program that is used to remove a bolt from the top of a box. A program such as this would be automatically generated by the master system as the operator moves the master manipulator into contact




(a) (b)

Figure 1: Impact wrench (a)surrounded by wrist, (b)removing bolt

with the bolt in the virtual remote site, and turns the master manipulator in the direction of unscrewing the bolt - an almost automatic action. Explanation and notes on the program will follow.

```
0.0   >> Execution Environment #0
0.1   UseTool(Wrench)
0.2   DefineVector(CP;<0.0,0.0,220.0>:EE)
0.3   DefineVector(X;<1.0,0.0,0.0>:EE)
0.4   DefineVector(Y;<0.0,1.0,0.0>:EE)
0.5   DefineTaskFrame(N:EE;CP;X;Y;?)
0.6   UseFrame(N)
0.7   AssignMode(P,P,P,P,P,P)
0.8   GuardForce(<0.0,0.0,-1.0>;<0.0,0.0,0.0>)
0.9   Move(5.0;<0.0,0.0,4.0>;<0.0,0.0,0.0>)
0.10  AssignMode(P,P,F,P,P,P)
0.11  Force(<0.0,0.0,1.0>;<0.0,0.0,0.0>)

1.0   >> Execution Environment #1
1.1   GuardVelocity(<0.0,0.0,1.0>;<0.0,0.0,0.0>)
1.2   Move(2.0;<0.0,0.0,0.0>;<0.0,0.0,-2.0>)
```

Execution environment #0 contains all the information needed to move the impact wrench into contact with the bolt, via a guarded move. After the impact wrench is designated for use, and a task frame is created with origin at the tool tip (see figure 1(b)) and the z-axis aligned with the rotation of the wrench, the commands generated are identical to a similar motion into contact without the use of a tool.

Execution environment #1 contains the necessary commands to remove the bolt. A GuardVelocity() command is used to monitor the outward velocity of the bolt. The rotation indicated in the Move() command turns the impact

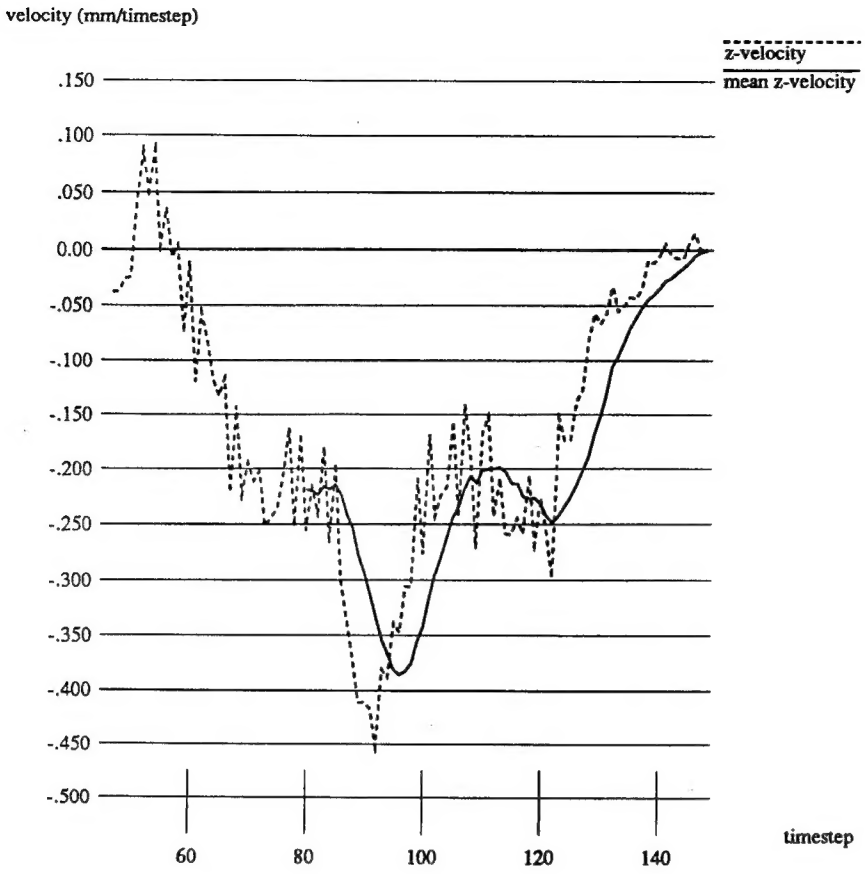


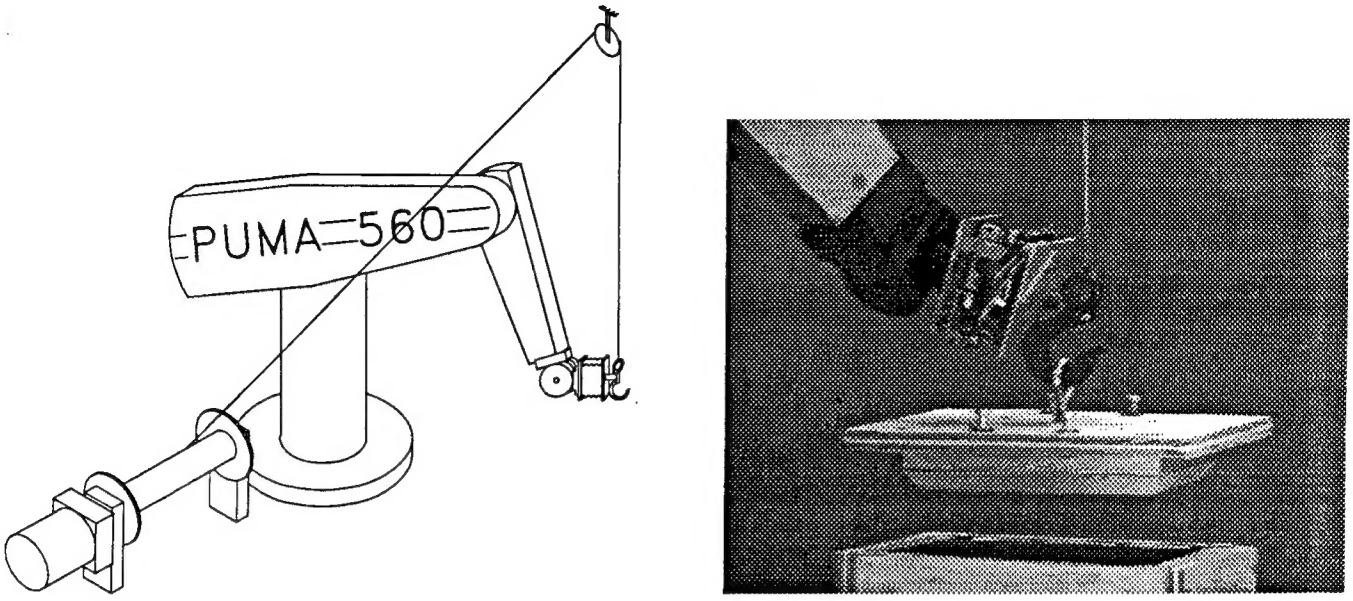

Figure 2: Velocity of robot while removing bolt

wrench on, and the nut is removed.

Figure 1(b) shows the robot-mounted impact wrench removing a bolt, as in the sample program above. The task frame coordinate system is indicated in the figure, with the origin at the tool tip.

In order to sense when the bolt is fully removed, the GuardVelocity() command is used. Shown in figure 2 is the actual sensed velocity (raw sensor data) in the direction of the bolt axis as the bolt is removed, as well as the mean velocity based on 10 time steps of data[2]. The mean velocity model is used by the system for detection of sensor events because of the noise level of the raw sensor data. Note that the velocity model is not started until 1/2 second into the motion. This is to allow for motion transients to abate. In this run, for example, the socket did not initially seat over the head of the bolt. When the impact wrench is turned on, the socket moves down over the bolt head, as indicated by the initial positive velocity in the raw sensor data; this data does not interfere with detection of bolt removal. As the robot complies with the force of the bolt, the feed rate of the bolt (determined by the bolt pitch) gives rise to a negative velocity in the robot. The mean velocity is monitored, and motion stops when the velocity returns to zero, indicating that the bolt is no longer feeding. A slight delay in sensing is caused as a result of the averaging of the velocity, but this does not adversely affect the task execution.

[2]Each time step is 20ms, the rate that the remote site computer communicates with the robot controller




(a) (b)

Figure 3: (a) System controlled winch, (b) winch removing box top

## 4.2 Winch

A winch can increase the load-carrying capacity of the robot manipulator. The need for power to offset the force of gravity on payloads, which usually accounts for a large fraction of the total power used by the robot, is eliminated. Because the payload capacity of a manipulator is usually inversely proportional to its accuracy, a robot with higher accuracy can be used for manipulation of heavy objects when a winch is used.

The winch is sensorless, and thus relies on the robot for sensing. The winch also adds a degree of freedom to the system: a prismatic joint in the world z-coordinate. Motion in this direction is naturally controlled by the winch when the winch is specified with the UseTool() command. The corresponding degree of freedom in the manipulator is force controlled with a force preload to keep the winch cable taut.

The winch used for this research, illustrated in figure 3(a) is small, and not much more powerful than the robot itself[3]. However, knowledge gained about using the winch in conjunction with the robot can be applied to much more powerful winches. The winch operates at a fixed speed, and therefore the motions at the operator's station must be constrained to this motion when the operator is using the winch.

When the command to use the winch is parsed, the following parameters are automatically set:

- The z-direction mode for the robot manipulator is set for force control. The manipulator will then conform to forces in the z-direction. This assumes that a task frame aligned with the kinematic base (world) coordinates is used.
- A force is set in the negative z-direction, to preload the winch cable (this keeps the cable taut).
- Gains and maximum allowable forces/velocities are changed to reflect the increase in noise associated with the constrained system.
- Motions in the positive z-direction are transformed to raise the winch. Similarly, motions in the negative z-direction lower the winch.

[3] The payload capacity of the winch is approximately 12 lbs.; the Puma robot payload is 5 lbs.

The following is a sample program that is used to insert the winch hook into an eyebolt on the top of a box (see figure 4). Explanation and notes on the program will follow.

```
0.0   >> Execution Environment #0
0.1   UseTool(Winch)
0.2   DefineVector(X;<0.0,1.0,0.0>:KB)
0.3   DefineVector(Z;<0.0,0.0,1.0>:KB)
0.4   DefineTaskFrame(TF:EE;WST;X;?;Z)
0.5   UseFrame(TF)
0.6   AssignMode(P,P,F,P,P,P)
0.7   GuardVelocity(<0.0,0.0,1.0>;<0.0,0.0,0.0>)
0.8   Move(9.0;<0.0,0.0,-15.0>;<0.0,0.0,0.0>)

1.0   >> Execution Environment #1
1.1   DefineVector(C;<0.0,30.0,160.0>:EE)
1.2   DefineTaskFrame(PP:EE;C;X;?;Z)
1.3   UseFrame(PP)
1.4   Move(1.5;<0.0,0.0,1.0>;<0.0,-0.4,0.0>)

2.0   >> Execution Environment #2
2.1   AssignMode(P,F,F,P,P,P)
2.2   GuardForce(<2.0,0.0,0.0>;<0.0,0.0,0.0>)
2.3   Move(5.0;<-12.0,0.0,0.0>;<0.0,0.0,0.0>)

3.0   >> Execution Environment #3
3.1   DefineVector(C;<0.0,-52.0,180.0>:EE)
3.2   DefineTaskFrame(PP:KB;C;X;?;Z)
3.3   UseFrame(PP)
3.4   Move(1.5;<0.0,0.0,0.0>;<0.0,-1.0,0.0>)

4.0   >> Execution Environment #4
4.1   Move(9.0;<0.0,0.0,15.0>;<0.0,0.0,0.0>)
```

This simple set of instructions is typical of a program that would be automatically generated by the master system from simple motions of the master arm by the human operator. Absent are any exploratory motions that may be necessary to identify the location of the eye on top of the box.

Each execution environment in the above program contains the commands needed to move between successive steps of figure 4. In execution environment #0, command 0.1 informs the remote system that the winch is being used. This implicitly results in a force applied in the negative z-direction. Commands 0.2 to 0.5 define a task frame to be used by the control. It is important to note that when using the winch, the z-direction of the task frame must correspond to the z-direction in kinematic base coordinates. Because the system will be moving, however, the frame itself is defined to be dynamic in command 0.4, by specifying the frame name TF to be with respect to the dynamic frame EE. After the frame has been designated, the commands are equivalent to commands generated without the use of a tool.

The robot is controlled in force mode in the z-direction with a force preload of approximately 2 lbs. on the winch cable. Any motion specified in the z-direction is assumed to have the velocity of the fixed-speed winch, and the winch

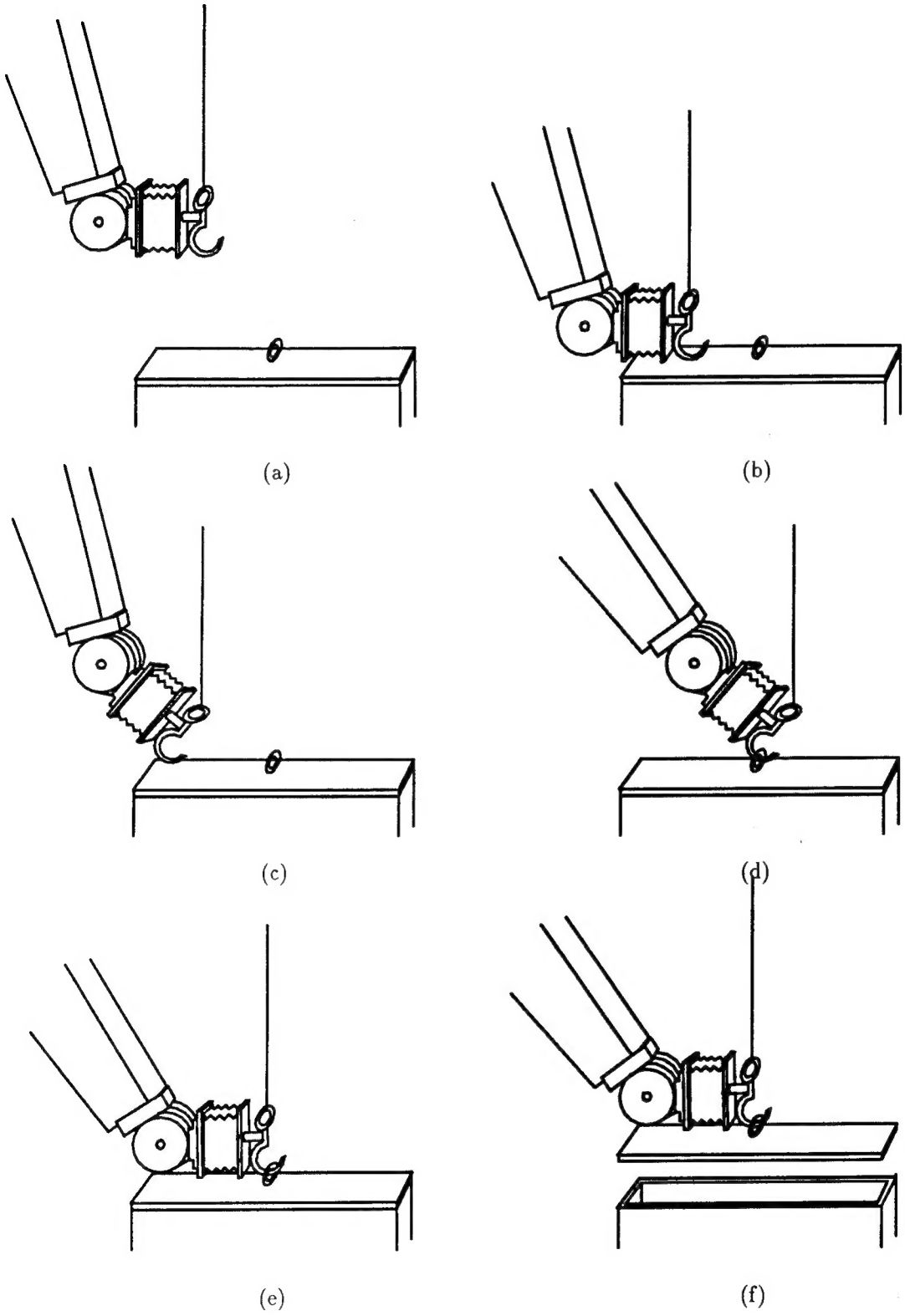



Figure 4: Winch experiment

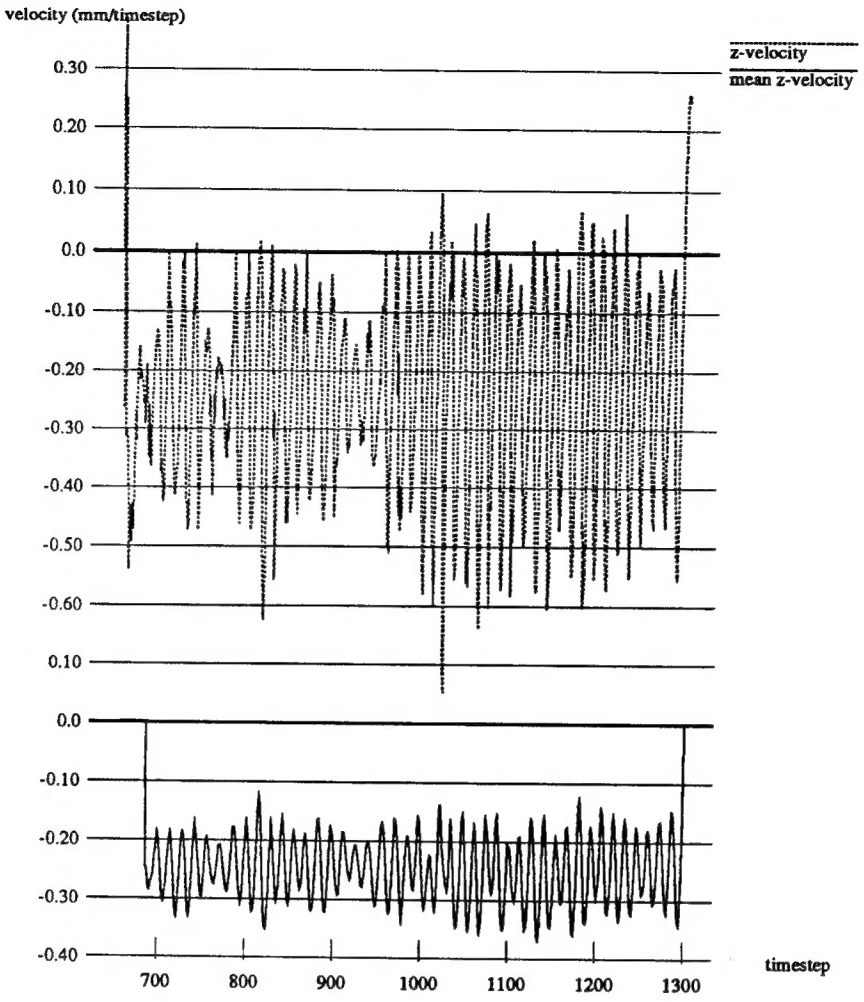



Figure 5: Hook velocity data for guarded move a–b

is controlled up or down as specified. The robot complies adequately to this motion. All other motion directions are controlled by the robot.

Sensing for the winch is more difficult than expected. Because the robot/tool system is constrained, internal forces are greater, and more noise is produced. However, the adaptive sensing algorithm is able to determine sensor events from the additional operating noise produced in this example.

Figure 5 shows the actual sensed velocity (raw data) and mean velocity (computed over 10 time steps) of the hook in the z-direction for the motion commanded in execution environment #1 above (notice that there are two separate scales on the y-axis of the graph, in order to make the data easier to read). The mean velocity model is not initialized until 1/2 second into the move, to avoid any irregular or transient data when the move begins. The velocity data is very noisy, and has a periodic frequency of approximately 3.3 Hz, which can be attributed the natural frequency of the wrist/winch-cable mechanical system. The mean velocity data reduces the amplitude of this vibration and delays the signal by about 1/10 second. Using the mean velocity data, the change in velocity caused by the collision with the box is easily discerned. The delay of 1/10 second is acceptable.

Figure 6 shows the implicit force data for the move into contact with the eyebolt. The data shown is in the direction of the motion. Note that the average force in the motion before contact (before approximately 755 timesteps on the x-axis of the graph) is non-zero. This is caused by internal forces built up by the constraints of the system. The

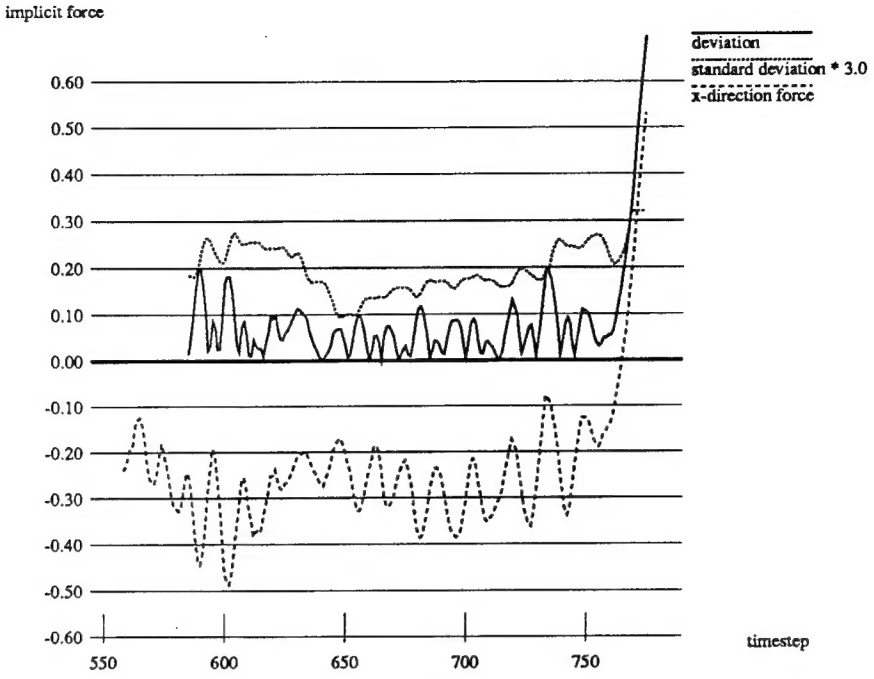



Figure 6: Hook force data for guarded move c–d

GuardForce command checks the deviation of the force data against the standard deviation model that is built up over the previous 30 time steps (.6 second). If the deviation of the force data is above 3 times the standard deviation for 4 consecutive time steps, a collision is detected. This rather stringent criteria is necessary in order to reject spurious, non-collision related data. It does, however, add slightly to the time delay in detecting collisions.

# 5 CONCLUSIONS

The methods presented here to control tools in a teleoperative system have the advantage that they do not increase the burden of work for the human operator. Except for the initial selection of a tool for use, the operation of tools is transparent to the operator. Because they do not require extra sensory input, the problems of sensor fusion and extra computational time needed to read and interpret additional signals are avoided, as well as avoiding the cost and implementation of another sensor in the system.

A host of questions involving tool usage were raised during this research and not addressed. Among these are:

- What exploration techniques are necessary to find the bolt/eye/etc?

This question was not addressed in the examples presented here. Obviously, with a tolerance limit larger than the size of the bolt (for example), some exploration will be necessary. This leads to another question:

- What tolerances are necessary to be successful at removing a bolt, mating a hook and eye, etc.?

If the initial tolerance limits, defined by the accuracy of the initial imaging of the remote system, are not accurate enough to find or make contact with the bolt, some method of refining the accuracy of the model will be necessary. Efforts to use data collected at the remote site to refine the graphical model are currently in progress.

These first two questions can be more easily researched when the tool implementation at the master site is operational.

Other questions that may be asked include:

- Is the tool control implemented here too tool-specific?
- Have we chosen tools that are general enough to show usage characteristics of all powered tools?

The experiments presented here use only two powered tools. However, the methodology for tool usage appears to applicable to a wide range of tools that the robot can use. However, there are certain end effectors, namely grippers and hands, that add extra degrees of freedom to the system that can not replace coincident degrees of freedom of the robot manipulator.

# 6 ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. BCS-89-01352, "Model-Based Teleoperation in the Presence of Delay." Any opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the National Science Foundation.

# 7 REFERENCES

[1] William R. Ferrell and Thomas B. Sheridan. Supervisory control of remote manipulation. *IEEE Spectrum*, 81–88, October 1967.

[2] Janez Funda. *Teleprogramming: Towards Delay-Invariant Remote Manipulation.* PhD thesis, University of Pennsylvania, 1991.

[3] Thomas Lindsay and Richard P. Paul. *Design of a Tool-Surrounding Compliant Instrumented Wrist.* Technical Report MS-CIS-91-30, GRASP LAB 258, University of Pennsylvania, April 1991.

[4] Richard P. Paul, Janez Funda, Thierry Simeon, and Thomas Lindsay. Teleprogramming for autonomous underwater manipulation systems. In *Intervention '90*, pages 91–95, The Marine Technology Society, June 1990.

[5] Peter M. Will and David D. Grossman. An experimental system for computer controlled mechanical assembly. *IEEE Transactions on Computers*, C-24(9):879–888, 1975.